19

United States Patent and Trademark Office

UNITED STATES DEPARTMENT OF COMMERCE
United States Patent and Trademark Office
Address: COMMISSIONER FOR PATENTS
P.O. Box 1450
Alexandria, Virginia 22313-1450
www.uspto.gov

APPLICATION NO.	FILING DATE	FIRST NAMED INVENTOR	ATTORNEY DOCKET NO.	CONFIRMATION NO.
09/752,796	12/29/2000	Adi Yoaz	42390P9574	9366

7590 01/12/2005

BLAKELY, SOKOLOFF, TAYLOR & ZAFMAN LLP
Seventh Floor
12400 Wilshire Boulevard
Los Angeles, CA 90025

EXAMINER	
GERSTL, SHANE F	
ART UNIT	PAPER NUMBER
2183	

DATE MAILED: 01/12/2005

Please find below and/or attached an Office communication concerning this application or proceeding.

Office Action Summary	**Application No.** 09/752,796	**Applicant(s)** YOAZ ET AL.	
	Examiner Shane F Gerstl	**Art Unit** 2183	

-- The MAILING DATE of this communication appears on the cover sheet with the correspondence address --

Period for Reply

A SHORTENED STATUTORY PERIOD FOR REPLY IS SET TO EXPIRE 3 MONTH(S) FROM THE MAILING DATE OF THIS COMMUNICATION.

- Extensions of time may be available under the provisions of 37 CFR 1.136(a). In no event, however, may a reply be timely filed after SIX (6) MONTHS from the mailing date of this communication.
- If the period for reply specified above is less than thirty (30) days, a reply within the statutory minimum of thirty (30) days will be considered timely.
- If NO period for reply is specified above, the maximum statutory period will apply and will expire SIX (6) MONTHS from the mailing date of this communication.
- Failure to reply within the set or extended period for reply will, by statute, cause the application to become ABANDONED (35 U.S.C. § 133). Any reply received by the Office later than three months after the mailing date of this communication, even if timely filed, may reduce any earned patent term adjustment. See 37 CFR 1.704(b).

Status

1)☒ Responsive to communication(s) filed on *26 November 2004*.

2a)☐ This action is **FINAL**. 2b)☒ This action is non-final.

3)☐ Since this application is in condition for allowance except for formal matters, prosecution as to the merits is closed in accordance with the practice under *Ex parte Quayle*, 1935 C.D. 11, 453 O.G. 213.

Disposition of Claims

4)☒ Claim(s) *1-26* is/are pending in the application.

4a) Of the above claim(s) _____ is/are withdrawn from consideration.

5)☐ Claim(s) _____ is/are allowed.

6)☒ Claim(s) *1-26* is/are rejected.

7)☐ Claim(s) _____ is/are objected to.

8)☐ Claim(s) _____ are subject to restriction and/or election requirement.

Application Papers

9)☐ The specification is objected to by the Examiner.

10)☐ The drawing(s) filed on _____ is/are: a)☐ accepted or b)☐ objected to by the Examiner.

Applicant may not request that any objection to the drawing(s) be held in abeyance. See 37 CFR 1.85(a).

Replacement drawing sheet(s) including the correction is required if the drawing(s) is objected to. See 37 CFR 1.121(d).

11)☐ The oath or declaration is objected to by the Examiner. Note the attached Office Action or form PTO-152.

Priority under 35 U.S.C. § 119

12)☐ Acknowledgment is made of a claim for foreign priority under 35 U.S.C. § 119(a)-(d) or (f).

a)☐ All b)☐ Some * c)☐ None of:

1.☐ Certified copies of the priority documents have been received.

2.☐ Certified copies of the priority documents have been received in Application No. _____.

3.☐ Copies of the certified copies of the priority documents have been received in this National Stage application from the International Bureau (PCT Rule 17.2(a)).

* See the attached detailed Office action for a list of the certified copies not received.

Attachment(s)

1) ☐ Notice of References Cited (PTO-892)

2) ☐ Notice of Draftsperson's Patent Drawing Review (PTO-948)

3) ☐ Information Disclosure Statement(s) (PTO-1449 or PTO/SB/08)
Paper No(s)/Mail Date _____.

4) ☐ Interview Summary (PTO-413)
Paper No(s)/Mail Date. _____ .

5) ☐ Notice of Informal Patent Application (PTO-152)

6) ☐ Other: _____.

DETAILED ACTION

1. Claims 1-26 have been examined.

Papers Received

2. Receipt is acknowledged of the request for continued examination and amendment papers submitted, where the papers have been placed of record in the file.

3. The 35 USC 112 rejections to the claims have been overcome by the amendment and are herein withdrawn.

Claim Rejections - 35 USC § 103

4. The following is a quotation of 35 U.S.C. 103(a) which forms the basis for all obviousness rejections set forth in this Office action:

> (a) A patent may not be obtained though the invention is not identically disclosed or described as set forth in section 102 of this title, if the differences between the subject matter sought to be patented and the prior art are such that the subject matter as a whole would have been obvious at the time the invention was made to a person having ordinary skill in the art to which said subject matter pertains. Patentability shall not be negatived by the manner in which the invention was made.

5. Claims 1-18 are rejected under 35 U.S.C. 103(a) as being unpatentable over Kahle (5,467,473) in view of Yoaz (5,987,595) and in view of Lipasti (On the Value Locality of Store Instructions).

6. In regard to claim 1,

a. Kahle discloses an apparatus comprising:

i. A processing section (figure 1);

ii. an extended load buffer; [Figure 5 gives a layout of a load queue which is also a buffer.]

iii. a marking processing section; [Figure 6, shows in step 4 that a load program number is placed or marked into the load queue (extended load buffer). Therefore a marking unit must exist to perform this marking.]

iv. a comparing processing section; [Figure 6, step 6 and column 2, lines 56-59 show that a store address is compared to the load addresses of the load buffer. For this comparison to take place a comparison unit must exist.]

v. and a recovery processing section; [Figure 6, steps 9 and 10 and column 3, lines 4-11 show that a load instruction must be placed in original order and reexecuted. It is shown that this is because of a conflict that exists and therefore original order must be recovered. This must be accomplished with a recovery unit.]

vi. wherein unexecuted load instructions are advanced over silent store instructions. [Column 2, lines 56-59 show that when a store executes, it's address is compared to previously executed load instructions, in a load queue, which executed out of order ahead of the store. So in the timeframe of the store execution the loads have been previously executed. However, in order to execute the load instructions out of order ahead of the store, the unexecuted load instructions are inherently advanced over the store for execution and once executed become previously executed loads that were previously advanced over the store. This is further show in column 3, lines 57-66. An example is in

Figure 2 with column 5, line 46 – column 6, line 13 where a load is executed ahead of a store instruction and thus the load is advanced ahead of a store instruction for execution (and thus was unexecuted when advanced). When the data being stored by the store instruction is the same as the data already in the register, the store instruction is a silent store and was likewise advanced over by the load. Since load instructions can be advanced past any store, they will also be advanced past silent stores, which do not affect system state as defined in the specification.]

b. Kahle does not disclose a predictor having a collision history table (CHT), said predictor for predicting silent store instructions, and that the processing section is coupled to the predictor. Kahle also does not disclose wherein the predictor compares an unexecuted load instruction with an issued and unexecuted store instruction value, and the unexecuted load instruction bypasses the issued store instruction for execution if the unexecuted load instruction value and the issued and unexecuted store value are the same.

c. Yoaz has disclosed a predictor having a collision history table (CHT) (figure 3, element 88). Column 3, lines 50-52 show that the CHT is used for predicting and thus is part of a predictor along with the control unit (figure 3, element 102). Figure 3 shows that the CHT or predictor is coupled to a recorder buffer, 94, using some control logic. Column 6, lines 35-36, shows that this buffer holds entries for load instructions. Yoaz has disclosed in column 3, lines 50-60 that the predictor is used for predicting load instructions so that loads can be executed ahead of stores. Lipasti discusses the notion of a silent store on page 183, column 2, last paragraph. It has the same definition as given by the applicant. Lipasti mentions on page 183, column 2, third paragraph that stride prediction is used. One will notice that in this same paragraph Lipasti speaks of silent stores and how a tagged last value predictor's limit is reached by these silent store and that the stride predictor is used by their design because it is not limited by silent stores and thus can more accurately predict stores including silent stores. Further proof of the use of silent store prediction is shown in section 3.4. It is shown that a "perfect method" is used to squash silent stores once they are predicted. The Examiner is taking Official notice that it is well known to one of ordinary skill in the art to compare an unexecuted load instruction with an unexecuted store instruction value, and have the unexecuted load instruction bypass the issued store instruction for execution if the unexecuted load instruction value and the issued and unexecuted store value are the same.

d. Yoaz has shown in column 2, lines 58-63 that his method is able to execute more load instructions out of order (based on the predictor) for faster processor operations. These faster processor operations would have motivated one of ordinary skill in the art to modify the design of Kahle to use the collision history table and predictor described by Yoaz. Page 185, section 3.1 of Lipasti then shows that squashing these silent stores (using prediction as shown in section 3.4 under the perfect method) allows a designer to obtain greater

> performance from existing structures, or a reduction in size or complexity of the system. This ability to obtain greater performance or reduction in size would have motivated one of ordinary skill in the art to modify the design of Kahle to include the silent store prediction given by Lipasti. With these modifications in place the design now predicts what loads can be advanced ahead of stores, effectively predicting silent stores as shown above. One of ordinary skill in the art would have recognized the advantages of comparing an unexecuted load instruction with an unexecuted store instruction value, and have the unexecuted load instruction bypass the issued store instruction for execution if the unexecuted load instruction value and the issued and unexecuted store value are the same so that a memory access is avoided and performance is sped up by advancing a load over a store (and squashing the store) while sending data directly from the register in the store instruction to the register destination of the load instruction.

It would have been obvious to one of ordinary skill in the art at the time of invention to modify the design of Kahle to include a predictor having a collision history table as disclosed by Yoaz that predicts silent stores as taught by Lipasti so that processor operations may be sped up and greater performance or smaller area may be realized. It would have been obvious to one of ordinary skill in the art to compare an unexecuted load instruction with an unexecuted store instruction value, and have the unexecuted load instruction bypass the issued store instruction for execution if the unexecuted load instruction value and the issued and unexecuted store value are the same so that a memory access is avoided and performance is sped up.

7. In regard to claim 2, Kahle in view of Yoaz and further in view of Lipasti has disclosed the apparatus of claim 1, wherein the predictor is a silent store predictor, as described above.

8. In regard to claim 3, Kahle in view of Yoaz and further in view of Lipasti has disclosed the apparatus of claim 2, as described above, wherein the silent store predictor uses path based indexing and the path is based on branches. [Column 5, lines 9-23 of Yoaz shows how the CHT (the predictor) is used. This section shows that the sequence of instructions is based on the correct prediction of branches. As shown in column 4, lines 8-10, the tag of the CHT is the linear instruction pointer. Thus, the predictor is indexed based on the linear sequence of instructions that is used based on branches.]

9. In regard to claim 4, Kahle in view of Yoaz and further in view of Lipasti has disclosed the apparatus of claim 3, wherein the silent store predictor is coupled with a state machine. [Column 4, lines 43-45 of Yoaz show that the CHT includes prediction bits being either sticky or saturating counters. A saturating counter in itself is a state machine because it varies its state or value based on inputs.]

10. In regard to claim 5, Kahle in view of Yoaz and further in view of Lipasti has disclosed the apparatus of claim 4, wherein the state machine is one of a 1-bit, 2-bit, and a sticky bit. [Column 4, lines 43-45 of Yoaz show that the CHT includes prediction bits being either sticky or saturating counters.]

11. In regard to claim 6, Kahle in view of Yoaz and further in view of Lipasti has disclosed the apparatus of claim 1, as described above, wherein the predictor is memory dependent. [Column 3, lines 54-60 of Yoaz show that predictor is based on memory addresses and thus is memory dependant.]

12. In regard to claim 7, Kahle in view of Yoaz and further in view of Lipasti has disclosed the apparatus of claim 1, wherein the extended load buffer comprises bit fields to mark load address match, load data match, load predict, and load flush, and bit fields for load address, load attribute and load data. [As described above, there is no reference in the specification for the elements 1110, 1130, 1140, and 1150: the load address match, load data match, and load flush, and load data. Therefore, these fields will be given a reasonable common English meaning. Also, the load attribute field is not defined explicitly and the same rule will be applied to it. As shown in figure 5, the extended load buffer holds a load address. This address is updated as a result of a load instruction or an instruction that was matched as a load. Therefore, this field is also the load address match field. Figure 5 also shows that the table includes a PC field, which gives the age of the instruction. This is load data of a load instruction, which is also a load attribute. Since the data is written there upon realizing that an instruction matches a load instruction, the field is also a load data match. Column 9, lines 10-21 show that a load can be marked upon a match indicating that the load must be re-executed. Since the extended load buffer holds information for the loads, it is clear that this buffer would then hold the marking bits in such an embodiment as the prediction bits. Since the marking bits set above are marked not only on a match of addresses but

also on improper ordering, these bits also signify a load flush because the load and subsequent instructions need to be flushed for re-execution as shown previously.]

13. In regard to claim 8, Kahle in view of Yoaz and further in view of Lipasti has disclosed the apparatus of claim 1, as described above, wherein the CHT is one of indexed by a tag and tagless. [Yoaz has shown in figure 2A a tagged CHT.]

14. In regard to claim 9, Kahle in view of Yoaz and further in view of Lipasti has disclosed the apparatus of claim 1, as described above, wherein the CHT includes distance bits. [Yoaz has shown in figure 2D a CHT including distance bits.]

15. In regard to claim 10,

a. Kahle discloses an apparatus comprising:

i. a processor (figure 1) having internal memory (figure 1, element 1);

ii. a bus coupled to the processor (figure 1, element 2);

iii. a memory coupled to a memory controller and the processor; [Column 2, line 65 – column 3, line 1 shows a memory used by and thus coupled to the processor. It is inherent that the memory has control logic so that it can be manipulated.]

iv. an extended load buffer; [Figure 5 gives a layout of a load queue which is also a buffer.]

v. a marking process; [Figure 6, shows in step 4 that a load program number is placed or marked into the load queue (extended load buffer). Therefore a marking unit must exist to perform this marking.]

vi. a comparing process; [Figure 6, step 6 and column 2, lines 56-59 show that a store address is compared to the load addresses of the load buffer. For this comparison to take place a comparison unit must exist.]

vii. and a recovery process; [Figure 6, steps 9 and 10 and column 3, lines 4-11 show that a load instruction must be placed in original order and reexecuted. It is shown that this is because of a conflict that exists and therefore original order must be recovered. This must be accomplished with a recovery unit.]

viii. wherein unexecuted load instructions are advanced over silent store instructions. [Column 2, lines 56-59 show that when a store executes, it's address is compared to previously executed load instructions, in a load queue, which executed out of order ahead of the store. So in the timeframe of the store execution the loads have been previously executed. However, in order to execute the load instructions out of order ahead of the store, the unexecuted load instructions are inherently advanced over the store for execution and once executed become previously executed loads that were previously advanced over the store. This is further show in column 3, lines 57-66. An example is in Figure 2 with column 5, line 46 – column 6, line 13 where a load is executed ahead of a store instruction and thus the load is advanced ahead of a store instruction for execution (and thus was unexecuted when advanced). When the data being stored by the store instruction is the same as the data already in the register, the store instruction is a silent store and was likewise advanced over by the load. Since load instructions can be advanced past any store, they will also be advanced past silent stores, which do not affect system state as defined in the specification.]

b. Kahle does not disclose a predictor having a collision history table (CHT), said predictor for predicting silent store instructions, or that the extended load buffer is coupled to the predictor. Kahle also does not disclose wherein the predictor compares an unexecuted load instruction with an issued and unexecuted store instruction value, and the unexecuted load instruction bypasses the issued store instruction for execution if the unexecuted load instruction value and the issued and unexecuted store value are the same.

c. Yoaz has disclosed a predictor having a collision history table (CHT) (figure 3, element 88). Column 3, lines 50-52 show that the CHT is used for predicting and thus is part of a predictor along with the control unit (figure 3, element 102). Figure 3 shows that the CHT or predictor is coupled to a recorder buffer, 94, using some control logic. Column 6, lines 35-36, shows that this buffer holds entries for load instructions. Yoaz has disclosed in column 3, lines 50-60 that the predictor is used for predicting load instructions so that loads can be executed ahead of stores. Lipasti discusses the notion of a silent store on page 183, column 2, last paragraph. It has the same definition as given by the applicant. Lipasti mentions on page 183, column 2, third paragraph that stride prediction is used. One will notice that in this same paragraph Lipasti speaks of silent stores and how a tagged last value predictor's limit is reached by these silent store and that the stride predictor is used by their design because it is not limited by silent stores and thus can more accurately predict stores including silent stores. Further proof of the use of silent store prediction is shown in section 3.4. It is shown that a "perfect method" is used to squash silent stores once they are predicted. The Examiner is taking Official notice that it is well known to one of ordinary skill in the art to compare an unexecuted load instruction with an unexecuted store instruction value, and have the unexecuted load instruction bypass the issued store instruction for execution if the unexecuted load instruction value and the issued and unexecuted store value are the same.

d. Yoaz has shown in column 2, lines 58-63 that his method is able to execute more load instructions out of order (based on the predictor) for faster processor operations. These faster processor operations would have motivated one of ordinary skill in the art to modify the design of Kahle to use the collision history table and predictor described by Yoaz. Page 185, section 3.1 of Lipasti then shows that squashing these silent stores (using prediction as shown in section 3.4 under the perfect method) allows a designer to obtain greater performance from existing structures, or a reduction in size or complexity of the system. This ability to obtain greater performance or reduction in size would have motivated one of ordinary skill in the art to modify the design of Kahle to include the silent store prediction given by Lipasti. With these modifications in

> place the design now predicts what loads can be advanced ahead of stores, effectively predicting silent stores as shown above. One of ordinary skill in the art would have recognized the advantages of comparing an unexecuted load instruction with an unexecuted store instruction value, and have the unexecuted load instruction bypass the issued store instruction for execution if the unexecuted load instruction value and the issued and unexecuted store value are the same so that a memory access is avoided and performance is sped up by advancing a load over a store (and squashing the store) while sending data directly from the register in the store instruction to the register destination of the load instruction.

It would have been obvious to one of ordinary skill in the art at the time of invention to modify the design of Kahle to include a predictor having a collision history table as disclosed by Yoaz that predicts silent stores as taught by Lipasti so that processor operations may be sped up and greater performance or smaller area may be realized. It would have been obvious to one of ordinary skill in the art to compare an unexecuted load instruction with an unexecuted store instruction value, and have the unexecuted load instruction bypass the issued store instruction for execution if the unexecuted load instruction value and the issued and unexecuted store value are the same so that a memory access is avoided and performance is sped up.In regard to claim 11, Kahle in view of Yoaz and further in view of Lipasti has disclosed the apparatus of claim 10, wherein the predictor is a silent store predictor, as described above.

16. In regard to claim 12, Kahle in view of Yoaz and further in view of Lipasti has disclosed the apparatus of claim 11, as described above, wherein the silent store predictor uses path based indexing and the path is based on branches. [Column 5, lines 9-23 of Yoaz shows how the CHT (the predictor) is used. This section shows that the sequence of instructions is based on the correct prediction of branches. As shown in column 4, lines 8-10, the tag of the CHT is the linear instruction pointer. Thus, the predictor is indexed based on the linear sequence of instructions that is used based on branches.]

17. In regard to claim 13, Kahle in view of Yoaz and further in view of Lipasti has disclosed the apparatus of claim 12, wherein the silent store predictor is coupled with a state machine. [Column 4, lines 43-45 of Yoaz show that the CHT includes prediction bits being either sticky or saturating counters. A saturating counter in itself is a state machine because it varies its state or value based on inputs.]

18. In regard to claim 14, Kahle in view of Yoaz and further in view of Lipasti has disclosed the apparatus of claim 13, wherein the state machine is one of a 1-bit, 2-bit, and a sticky bit. [Column 4, lines 43-45 of Yoaz show that the CHT includes prediction bits being either sticky or saturating counters.]

19. In regard to claim 15, Kahle in view of Yoaz and further in view of Lipasti has disclosed the apparatus of claim 10, as described above, wherein the predictor is memory dependent. [Column 3, lines 54-60 of Yoaz show that predictor is based on memory addresses and thus is memory dependant.]

20. In regard to claim 16, Kahle in view of Yoaz and further in view of Lipasti has disclosed the apparatus of claim 10, wherein the extended load buffer comprises bit fields to mark load address match, load data match, load predict, and load flush, and bit fields for load address, load attribute and load data. [As described above, there is no reference in the specification for the elements 1110, 1130, 1140, and 1150: the load address match, load data match, and load flush, and load data. Therefore, these fields will be given a reasonable common English meaning. Also, the load attribute field is not defined explicitly and the same rule will be applied to it. As shown in figure 5, the extended load buffer holds a load address. This address is updated as a result of a load instruction or an instruction that was matched as a load. Therefore, this field is also the load address match field. Figure 5 also shows that the table includes a PC field, which gives the age of the instruction. This is load data of a load instruction, which is also a load attribute. Since the data is written there upon realizing that an instruction matches a load instruction, the field is also a load data match. Column 9, lines 10-21 show that a load can be marked upon a match indicating that the load must be re-executed. Since the extended load buffer holds information for the loads, it is clear that this buffer would then hold the marking bits in such an embodiment as the prediction bits. Since the marking bits set above are marked not only on a match of addresses but also on improper ordering, these bits also signify a load flush because the load and subsequent instructions need to be flushed for re-execution as shown previously.]

21. In regard to claim 17, Kahle in view of Yoaz and further in view of Lipasti has disclosed the apparatus of claim 10, as described above, wherein the CHT is one of indexed by a tag and tagless. [Yoaz has shown in figure 2A a tagged CHT.]

22. In regard to claim 18, Kahle in view of Yoaz and further in view of Lipasti has disclosed the apparatus of claim 10, as described above, wherein the CHT includes distance bits. [Yoaz has shown in figure 2D a CHT including distance bits.]

23. Claims 19-26 are rejected under 35 U.S.C. 103(a) as being unpatentable over Kahle in view of Lipasti (On the Value Locality of Store Instructions).

24. In regard to claim 19,

a. Kahle discloses a method comprising:

i. fetching an instruction (figure 6, step 1) and determining if an instruction is one of a store and a load (figure 6, step 3);

ii. issuing the store instruction; [Figure 6, step 5 shows that stores are executed and thus must by issued in the pipeline.]

iii. comparing an address and data of the store address with load instructions in an extended load buffer (figure 6, steps 6 and 8); [Figure 5 gives a layout of the load queue used in figure 6 to hold load information. This queue is also a buffer. The PC value indicates a program number for comparison of age of the instructions and thus is data of the instructions.]

iv. setting marking bits in the extended load buffer if a match is found in the comparing; [Column 9, lines 10-21 show that a load can be marked upon a match indicating that the load must be re-executed. Since the extended load buffer holds information for the loads, it is clear that this buffer would then hold the marking bits in such an embodiment.]

v. updating a memory with store instruction if the store instruction can be retired; [Column 6, lines 37-39 show that the memory is updated when the result of a store is committed or retired.]

vi. and bypassing a store instruction and executing the load instruction ahead of the silent store instruction. [Column 2, lines 56-59 show that when a store executes, it's address is compared to previously executed load instructions, in a load queue, which executed out of order ahead of the store. So in the timeframe of the store execution the loads have been previously executed. However, in order to execute the load instructions out of order ahead of the store, the unexecuted load instructions are inherently advanced over the store for execution and once executed become previously executed loads that were previously advanced over the store. This is further show in column 3, lines 57-66. An example is in Figure 2 with column 5, line 46 – column 6, line 13 where a load is executed ahead of a store instruction and thus the load is advanced ahead of a store instruction for execution (and thus was unexecuted when advanced). When the data being stored by the store instruction is the same as the data already in the register, the store instruction is a silent store and was likewise advanced over by the load. Since load instructions can be advanced past any store, they will also be advanced past silent stores, which do not affect system state as defined in the specification.]

b. Kahle does not disclose performing a silent store prediction if the instruction is a store nor does Kahle disclose bypassing a predicted silent store if an unexecuted load instruction value matches the issued and unexecuted store instruction value;

c. Lipasti discusses the notion of a silent store on page 183, column 2, last paragraph. It has the same definition as given by the applicant. Lipasti mentions on page 183, column 2, third paragraph that stride prediction is used. One will notice that in this same paragraph Lipasti speaks of silent stores and how a tagged last value predictor's limit is reached by these silent store and that the stride predictor is used by their design because it is not limited by silent stores and thus can more accurately predict stores including silent stores. Further proof of the use of silent store prediction is shown in section 3.4. It is shown that a "perfect method" is used to squash silent stores once they are predicted. The Examiner is taking Official notice that it is well known to one of ordinary skill in the art to compare an unexecuted load instruction with an unexecuted store instruction value, and have the unexecuted load instruction bypass the issued store instruction for execution if the unexecuted load instruction value and the issued and unexecuted store value are the same.

d. Page 185, section 3.1 of Lipasti then shows that squashing these silent stores (using prediction as shown in section 3.4 under the perfect method) allows a designer to obtain greater performance from existing structures, or a reduction in size or complexity of the system. This ability to obtain greater performance or reduction in size would have motivated one of ordinary skill in the art to modify the design of Kahle to include the silent store prediction given by Lipasti. With these modifications in place the design now predicts what loads can be advanced ahead of stores, effectively predicting silent stores as shown above. One of ordinary skill in the art would have recognized the advantages of comparing an unexecuted load instruction with an unexecuted store instruction value, and have the unexecuted load instruction bypass the issued store instruction for execution if the unexecuted load instruction value and the issued and unexecuted store value are the same so that a memory access is avoided and performance is sped up by advancing a load over a store (and squashing the store) while sending data directly from the register in the store instruction to the register destination of the load instruction.

It would have been obvious to one of ordinary skill in the art at the time of invention to modify the design of Kahle to predict silent stores as taught by Lipasti so that processor operations may be sped up and greater performance or smaller area may be realized. It would have been obvious to one of ordinary skill in the art to compare an unexecuted load instruction with an unexecuted store instruction value, and have the unexecuted load instruction bypass the issued store instruction for execution if the unexecuted load instruction value and the issued and unexecuted store value are the same so that a memory access is avoided and performance is sped up.

25. In regard to claim 20, Kahle in view of Lipasti has disclosed the method of claim 19, as described above, further comprising preparing the load instruction for retirement, if the load instruction is complete, and determining if the load instruction is marked flush in the extended load buffer. [Column 6, lines 20-24 of Kahle show that a load is committed or retired. In preparation for this, the address and program number are removed from the load queue. Since the marking bits set above are marked not only on a match of addresses but also on improper ordering, these bits also signify a load flush because the load and subsequent instructions need to be flushed for re-execution as shown previously.]

26. In regard to claim 22, Kahle in view of Lipasti discloses the method of claim 19, wherein the memory is a cache. [As shown in column 6, lines 37-39 of Kahle, the completed store operation writes to a memory via a cache, thus the operation writes to the cache memory as well as a memory.]

27. In regard to claim 23,

a. Kahle discloses a program storage device readable by a machine comprising instructions that cause the machine to:

i. fetch an operation (figure 6, step 1) and determining if an instruction is one of a store instruction and a load instruction (figure 6, step 3);

ii. execute the store instruction (figure 6, step 5);

iii. compare an address and data of the store operation with load operations in an extended load buffer (figure 6, steps 6 and 8); [Figure 5 gives a layout of the load queue used in figure 6 to hold load information. This queue is also a buffer. The PC value indicates a program number for comparison of age of the instructions and thus is data of the instructions.]

iv. setting marking bits in the extended load buffer if a match is found in the compare instruction; [Column 9, lines 10-21 show that a load can be marked upon a match indicating that the load must be re-executed. Since the extended load buffer holds information for the loads, it is clear that this buffer would then hold the marking bits in such an embodiment.]

v. update a memory with store operation if the store operation can be retired; [Column 6, lines 37-39 show that the memory is updated when the result of a store is committed or retired.]

vi. and bypass a silent store operation and execute a load operation ahead of the silent store operation if the operation is a load. [Column 2, lines 56-59 show that when a store executes, it's address is compared to previously executed load instructions, in a load queue, which executed out of order ahead of the store. So in the timeframe of the store execution the loads have been previously executed. However, in order to execute the load instructions out of order ahead of the store, the unexecuted load instructions are inherently advanced over the store for execution and once executed become previously executed loads that were previously advanced over the store. This is further show in column 3, lines 57-66. An example is in Figure 2 with column 5, line 46 – column 6, line 13 where a load is executed ahead of a store instruction and thus the load is advanced ahead of a store instruction for execution (and thus was unexecuted when advanced). When the data being stored by the store instruction is the same as the data already in the register, the store instruction is a silent store and was likewise advanced over by the load. Since load instructions can be advanced past any store, they will also be advanced past silent stores, which do not affect system state as defined in the specification.]

b. Kahle does not disclose performing a silent store prediction if the operation is a store instruction nor does Kahle disclose bypassing a predicted silent store if an unexecuted load instruction value matches the issued and unexecuted store instruction value;

c. Lipasti discusses the notion of a silent store on page 183, column 2, last paragraph. It has the same definition as given by the applicant. Lipasti mentions on page 183, column 2, third paragraph that stride prediction is used. One will notice that in this same paragraph Lipasti speaks of silent stores and how a tagged last value predictor's limit is reached by these silent store and that the stride predictor is used by their design because it is not limited by silent stores and thus can more accurately predict stores including silent stores. Further proof of the use of silent store prediction is shown in section 3.4. It is shown that a "perfect method" is used to squash silent stores once they are predicted. The Examiner is taking Official notice that it is well known to one of ordinary skill in the art to compare an unexecuted load instruction with an unexecuted store instruction value, and have the unexecuted load instruction bypass the issued store instruction for execution if the unexecuted load instruction value and the issued and unexecuted store value are the same.

d. Page 185, section 3.1 of Lipasti then shows that squashing these silent stores (using prediction as shown in section 3.4 under the perfect method) allows a designer to obtain greater performance from existing structures, or a reduction in size or complexity of the system. This ability to obtain greater performance or reduction in size would have motivated one of ordinary skill in the art to modify the design of Kahle to include the silent store prediction given by Lipasti. With these modifications in place the design now predicts what loads can be advanced ahead of stores, effectively predicting silent stores as shown above. One of ordinary skill in the art would have recognized the advantages of comparing an unexecuted load instruction with an unexecuted store instruction value, and have the unexecuted load instruction bypass the issued store instruction for execution if the unexecuted load instruction value and the issued and unexecuted store value are the same so that a memory access is avoided and performance is sped up by advancing a load over a store (and squashing the store) while sending data directly from the register in the store instruction to the register destination of the load instruction.

It would have been obvious to one of ordinary skill in the art at the time of invention to modify the design of Kahle to predict silent stores as taught by Lipasti so that processor operations may be sped up and greater performance or smaller area may be realized. It would have been obvious to one of ordinary skill in the art to compare an unexecuted load instruction with an unexecuted store instruction value, and have the unexecuted load instruction bypass the issued store instruction for execution if the unexecuted load instruction value and the issued and unexecuted store value are the same so that a memory access is avoided and performance is sped up.

28. In regard to claim 24, Kahle in view of Lipasti has disclosed the method of claim 23, as described above, wherein the instructions further cause the machine to prepare the load operation for retirement if the load operation is complete, and determining if the load operation is marked flush in the extended load buffer. [Column 6, lines 20-24 of Kahle show that a load is committed or retired. In preparation for this, the address and program number are removed from the load queue. Since the marking bits set above are marked not only on a match of addresses but also on improper ordering, these bits also signify a load flush because the load and subsequent instructions need to be flushed for re-execution as shown previously.]

29. In regard to claim 26, Kahle in view of Lipasti discloses the program storage device of claim 23, wherein the memory is a cache. [As shown in column 6, lines 37-39 of Kahle, the completed store operation writes to a memory via a cache, thus the operation writes to the cache memory as well as a memory.]

30. Claims 21 and 25 are rejected under 35 U.S.C. 103(a) as being unpatentable over Kahle in view of Lipasti as applied to claims 19-20, 22-24, and 26 above, and further in view of Yoaz.

31. In regard to claim 21,

a. Kahle in view of Lipasti has disclosed the method of claim 19, as shown above,

b. Kahle in view of Lipasti does not disclose wherein the predicting includes marking bits in a collision history table (CHT).

c. Yoaz has disclosed a wherein the predicting includes marking bits in a collision history table (CHT) (figure 3, element 88). Column 3, lines 50-52 show that the CHT is used for predicting. Column 5, lines 57-67, show updating or marking the CHT.

d. Yoaz has shown in column 2, lines 58-63 that his method is able to execute more load instructions out of order (based on the predictor) for faster processor operations. These faster processor operations would have motivated one of ordinary skill in the art to modify the design of Kahle in view of Lipasti to use the collision history table and predictor described by Yoaz.

It would have been obvious to one of ordinary skill in the art at the time of invention to modify the design of Kahle in view of Lipasti to include the collision history table predictor disclosed by Yoaz so that processor operations may be sped up.

32. In regard to claim 25,

a. Kahle in view of Lipasti has disclosed the method of claim 23, as shown above,

b. Kahle in view of Lipasti does not disclose wherein the instruction that causes the machine to predict silent stores includes an instruction that causes the machine to mark bits in a collision history table (CHT).

c. Yoaz has disclosed a wherein the instruction that causes the machine to predict silent stores includes an instruction that causes the machine to mark bits in a collision history table (CHT) (figure 3, element 88). Column 3, lines 50-52 show that the CHT is used for predicting. Column 5, lines 57-67, show updating or marking the CHT.

d. Yoaz has shown in column 2, lines 58-63 that his method is able to execute more load instructions out of order (based on the predictor) for faster processor operations. These faster processor operations would have motivated one of ordinary skill in the art to modify the design of Kahle in view of Lipasti to use the collision history table and predictor described by Yoaz.

It would have been obvious to one of ordinary skill in the art at the time of invention to modify the design of Kahle in view of Lipasti to include the collision history table predictor disclosed by Yoaz so that processor operations may be sped up.

Response to Arguments

33. Applicant's arguments have been fully considered but they are not persuasive.

34. Applicant has argued that the prior art of record does not disclose the limitation of claims 1, 10, 19, and 23 requiring that unexecuted load instructions are advanced over silent store instructions. Column 2, lines 56-59 show that when a store executes, it's address is compared to previously executed load instructions, in a load queue, which

executed out of order ahead of the store. So in the timeframe of the store execution the loads have been previously executed. However, in order to execute the load instructions out of order ahead of the store, the unexecuted load instructions are inherently advanced over the store for execution and once executed become previously executed loads that were previously advanced over the store. This is further show in column 3, lines 57-66. An example is in Figure 2 with column 5, line 46 – column 6, line 13 where a load is executed ahead of a store instruction and thus the load is advanced ahead of a store instruction for execution (and thus was unexecuted when advanced). When the data being stored by the store instruction is the same as the data already in the register, the store instruction is a silent store and was likewise advanced over by the load. Since load instructions can be advanced past any store, they will also be advanced past silent stores, which do not affect system state as defined in the specification.

35. Applicant has argued on pages that the prior art of record does not teach silent store prediction as disclosed in claims 1, 10, 19, and 23. Lipasti discusses the notion of a silent store on page 183, column 2, last paragraph. It has the same definition as given by the applicant. Lipasti mentions on page 183, column 2, third paragraph that stride prediction is used. One will notice that in this same paragraph Lipasti speaks of silent stores and how a tagged last value predictor's limit is reached by these silent store and that the stride predictor is used by their design because it is not limited by silent stores and thus can more accurately predict stores including silent stores. Further proof of the use of silent store prediction is shown in section 3.4. It is shown that a

"perfect method" is used to squash silent stores once they are predicted. Page 185, section 3.1 of Lipasti then shows that squashing these silent stores (using prediction as shown in section 3.4 under the perfect method) allows a designer to obtain greater performance from existing structures, or a reduction in size or complexity of the system. This ability to obtain greater performance or reduction in size would have motivated one of ordinary skill in the art to modify the design of Kahle to include the silent store prediction given by Lipasti. With these modifications in place the design now predicts what loads can be advanced ahead of stores, effectively predicting silent stores as shown above. Further, the 35 USC 103 modifications are proper since the prediction of silent stores is disclosed with proper motivation as given above.

36. The Applicant also argues that The Examiner does not disclose the new limitations of the independent claims, which are addressed above in the rejections.

Conclusion

37. The following is text cited from 37 CFR 1.111(c): In amending in reply to a rejection of claims in an application or patent under reexamination, the applicant or patent owner must clearly point out the patentable novelty which he or she thinks the claims present in view of the state of the art disclosed by the references cited or the objections made. The applicant or patent owner must also show how the amendments avoid such references or objections.

Any inquiry concerning this communication or earlier communications from the examiner should be directed to Shane F Gerstl whose telephone number is (571) 272-4166. The examiner can normally be reached on M-F 6:45-4:15 (First Friday Off).

If attempts to reach the examiner by telephone are unsuccessful, the examiner's supervisor, Eddie Chan can be reached on (571) 272-4162. The fax phone number for the organization where this application or proceeding is assigned is 703-872-9306.

Information regarding the status of an application may be obtained from the Patent Application Information Retrieval (PAIR) system. Status information for published applications may be obtained from either Private PAIR or Public PAIR. Status information for unpublished applications is available through Private PAIR only. For more information about the PAIR system, see http://pair-direct.uspto.gov. Should you have questions on access to the Private PAIR system, contact the Electronic Business Center (EBC) at 866-217-9197 (toll-free).

Shane F Gerstl
Examiner
Art Unit 2183

SFG
January 7, 2005

EDDIE CHAN
SUPERVISORY PATENT EXAMINER
TECHNOLOGY CENTER 2100